Recent Results in Software Process Modeling

Ray Madachy, Ph.D.
C-bridge Internet Solutions
University of Southern California Center for Software Engineering
rmadachy@c-bridge.com, madachy@usc.edu

1 Introduction

Understanding software process interactions and feedback is increasingly important given changing software development and evolution paradigms. Towards this end, a new graduate course in *Software Process Modeling* was developed by this author at the University of Southern California Center for Software Engineering (USC-CSE) [1]. It was first offered in the Fall of 1999, and the term projects were original investigations into critical process issues.

This abstract summarizes the student research projects, and highlights some results in terms of identifying important feedback in software processes. Rather than focus on previous contributions [2], [3], only new and previously unreported student work is described herein. The in-process book *Software Process Dynamics* [4] was the primary text for the class. Some of the student work incorporates new concepts in the book, such as inter-phase iterative feedback, process concurrence, personnel factors, learning feedback and model calibration techniques.

2 Course Summary

The course overviews the field of software process modeling, and addresses current research issues with student simulation projects. It is designed for students and software engineering professionals who are interested in understanding the dynamics of software development and assessing process strategies. Process modeling techniques for both continuous systems and discrete systems are covered, with a concentration in system dynamics modeling (continuous systems).

Examples of process and project dynamics covered are Rapid Application Development (RAD), the effects of schedule pressure, experience, global feedback and evolution, work methods such as reviews and quality assurance activities, task underestimation, bureaucratic delays, demotivating events, process concurrence, other socio-technical phenomena and the feedback therein. These complex and interacting process effects are modeled with system dynamics using continuous quantities interconnected in loops of information feedback and circular causality.

The course demonstrates how knowledge of the interrelated technical and social factors coupled with simulation tools can provide a means for software process improvement. More information can be found in [1].

3 Simulation Term Projects

Students were instructed in a fairly rigorous process for developing their models, starting with definitive statements of modeling goals and identifying reference behavior. Students were also asked to interview seasoned experts in their respective areas, develop surveys as appropriate, and pointers to real world data were provided. Group reviews and extensive validation tests were performed, and a standard format was used for the reports.

Students were allowed to do individual projects or have teams of two. Several of the projects employed teams. The five term projects investigated the following issues:

- the dynamics of architecture development in the inception and elaboration phases of the Model-Based Architecting and Software Engineering (MBASE) process
- COTS glue code development and integration dynamics
- reuse and language-level effects in software development
- CMM-based Software Process Improvement (SPI) strategies
- application of RAD techniques to pre-IPO Internet companies.

Table 1 categorizes the projects in terms of their high-level scope and purpose.

Table 1: Project Characterizations

Scope / Purpose	Portion of lifecycle	Development project	Long-term organization
Planning and control	MBASE Architecting	Reuse and High Level Languages Internet RAD COTS Glue Code	CMM Software Process Improvement
Process improvement and technology adoption			

Subsequent sections provide more highlights on the projects. The more in-depth studies performed by two-person teams are described first. Very few figures are included here due to limited space. In order to best visualize the feedback loops in the models, the reader is encouraged to read the original reports at [1].

3.1 MBASE Architecting

MBASE is an integrated software development approach developed at USC-CSE. It is used in departmental Software Engineering courses. The overall goals of the MBASE architecting study were to:

- investigate the dynamics of architecture development during early MBASE lifecyle phases
- identify the nature of process concurrence in early MBASE phases
- understand the impact of collaboration and prototyping on lifecycle parameters.

Some features of the model include:

- schedule as independent variable
- iterative process structures
- sequentiality and concurrency of activities
- phases - requirements and architecture/design
- activities - initial completion, coordination, quality assurance, iteration
- demand-based resource allocation
- external and internal precedence constraints
- calibration to CS577A project data (CS577A is a graduate course in Software Engineering taught by Dr. Barry Boehm where teams develop multimedia applications for internal USC library usage).

A high-level overview of the lifecycle artifact flows and feedback is shown in Figure 1, and the major causal loops are shown in Figure 2.

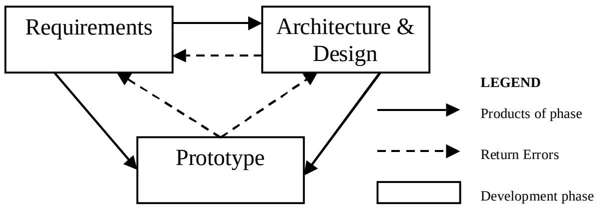

Figure 1: MBASE Lifecycle Artifact Relationships

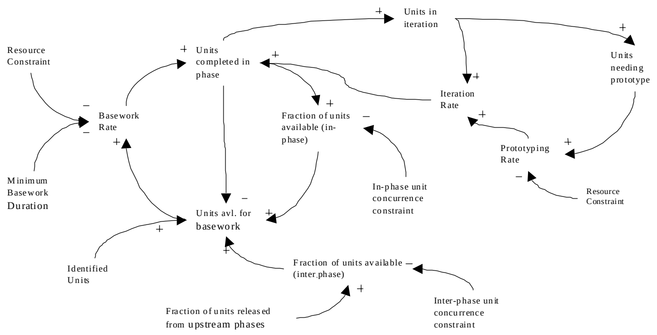
Units in iteration
Resource Constraint
Units completed in phase
Units needing prototype
Iteration Rate
Basework Rate
Fraction of units available (in-phase)
Prototyping Rate
Minimum Basework Duration
In-phase unit concurrence constraint
Resource Constraint
Units avl. for basework
Identified Units
Fraction of units available (inter.phase)
Fraction of units released from upstream phases
Inter-phase unit concurrence constraint

Figure 2: MBASE Architecting Causal Loop Diagram

This study fleshed out various prototyping feedback effects, inter-phase iteration feedback, and the model was calibrated to actual MBASE project data.

3.2 COTS Glue Code

This study investigated the overall lifecycle of incorporating COTS into a product, with a primary focus on the integration dynamics. Overall goals were to:

- understand glue code development, the COTS integration process and their correlation
- determine efficient starting points of glue code development and COTS integration
- calibrate the component parameters from COCOTS
- analyze the impact of new parameters such as ratio of new and updated COTS component and number of COTS component.

Figure 3 shows some of the feedback interactions in the model.

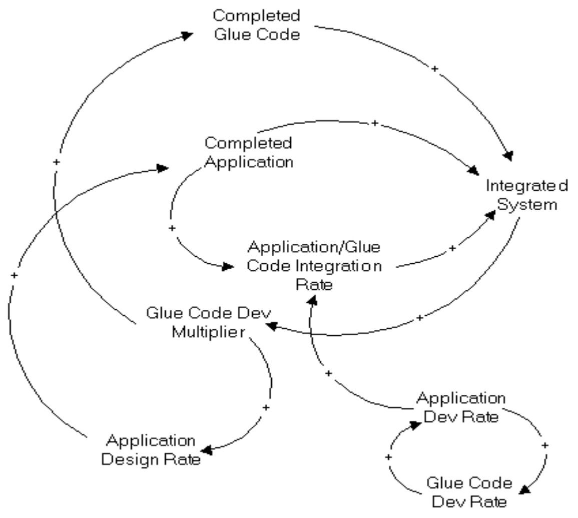

Figure 3: COT Glue Code Causal Loop Diagram

This study produced valuable results in terms of COTS and glue code integration planning.

3.3 Reuse and High Level Language

This study looked at dynamics of reuse as well as the effects of employing high-level languages. Goals for the project included:

- investigating project reuse dynamics
- productivity and effort of individual phases
- understand the effects of different language levels.

Some features of the model are

- rework is included
- learning curve formulations
- increased training for higher level languages.

See Figure 4 for the primary effects in the model. This work produced a validated model containing reuse and language effects, including a rate-based representation of reuse impact on project size that can serve as an archetype structure.

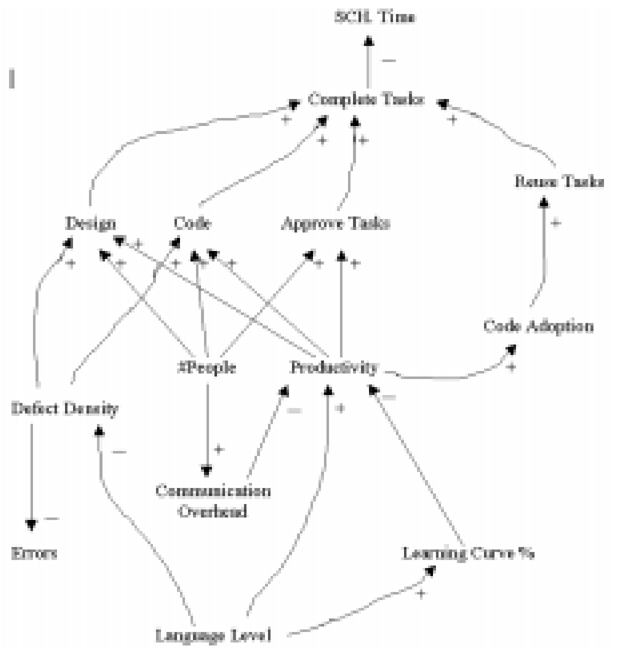

Figure 4: Reuse and High Level Language Causal Loop Diagram

3.4 CMM-Based Software Process Improvement

This work investigated software process improvement based on the CMM. Subgoals included the following:

- provide insight into complex process behavior
- help evaluate different approaches for improvement
- support planning, tracking and prediction
- reduce costs
- reduce cycle time
- reduce defects

An existing model [5] was adapted for use at Xerox. It was based on the scenario of a Xerox development group working from just assessed as a Level 2 organization moving towards achieving Level 3. Figure 5 shows the model at a high level, including feedback loops.

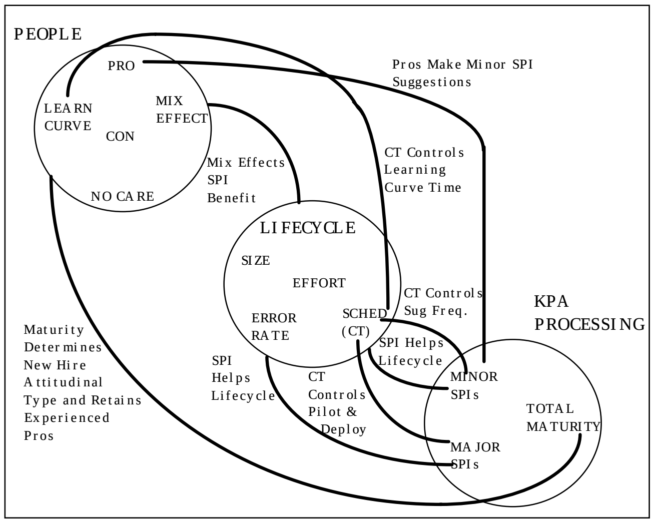

Figure 5: SPI Model Feedback Relationships

Referring to the figure, the ***Lifecycle*** process models how software size, effort, quality, and schedule relate to each other in order to produce a product. SPI benefits are modeled as percent reductions in either size, effort, error rate or schedule.

In ***People***, three attitudes of staff that affected the potential benefit of process improvement: pro-SPI people, con-SPI people, and no-care people. The attitudinal mix and the pro/con ratio can affect the overall potential benefit realized by a SPI effort.

KPA Processing models the timing of the flow of process improvements into the lifecycle and people subsystems.

The following causal loop description is the basis of the simulation model.

1. Major software process improvement (SPI) efforts are piloted and deployed based on project cycle time (i.e., pilot on one project, tailor and deploy on the subsequent project).
2. Major SPIs increase maturity (the probability of successfully achieving your goals).
3. Increased maturity attracts new hires and retains experienced staff that are "pro SPI" (i.e., they support and participate in SPI activities and are attracted to success and innovation).
4. Pro-SPI staff make minor SPI suggestions.
5. Major and minor SPIs decrease cycle time.
6. Decreased cycle time enables more major and minor SPIs to be accomplished.
7. Go back to 1 and repeat the cycle.

This model was calibrated for the Xerox environment, and the results are being used for internal SPI planning.

3.5 Internet RAD

The goals for this project included:

- investigating the dynamics of pre-IPO Internet companies
- contrasting the dynamics to non-Internet software development
- surveying companies and determine major impact factors.

Some features of the Internet RAD model are:

- a modified evolutionary delivery lifecycle with small teams
- schedule minimization
- outsourcing considerations
- defect detection and elimination
- short term and long-term feedback
- Internet preview and web-site personalization
- model sectors - Specification and Design, Outsourcing, Development, Integration and Personalization, Human Resources.

Short-term and long-term feedback to the web site integration can seen in the Integration and Personalization sector in Figure 6.

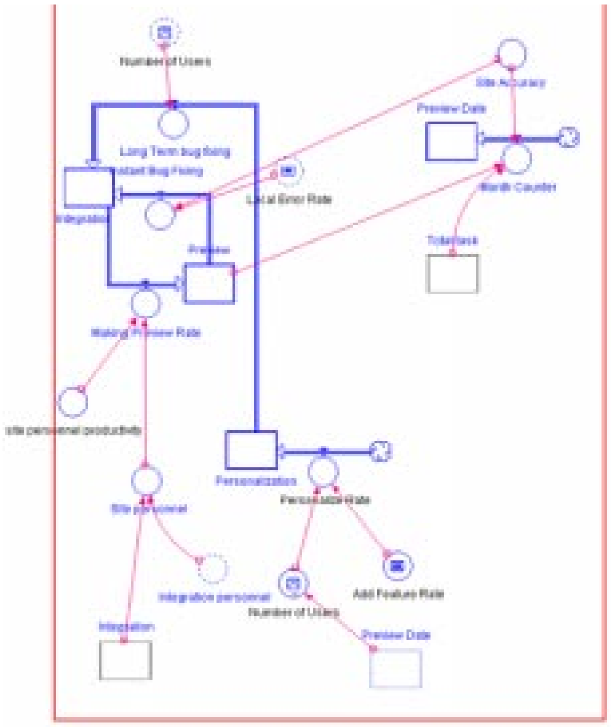

Figure 6: Internet RAD Integration and Personalization Sector

Contributions of this study include the lifecycle definition for Internet-unique instant software delivery, and identification of the two inherent types of feedback.

4 Feedback Summary

A summary of the most important feedback effects from these studies are shown in Table 2. Refer to the original papers for specific attributes of the feedback.

Table 2: Summary of Major Feedback

Project	Major Feedback Effects
MBASE Architecting	Feedback from prototype evaluation that impacts the requirements and architecture. Inter-phase iterative feedback from approval activities, reviews and testing. Reinforcing feedback from initially completed items. Reinforcing feedback from prototyping to iteration rate.
COTS Glue Code	Connected feedback loops between application development, glue code development and system integration.
Reuse and High Level Languages	Learning feedback effects on development and reuse rates. Professor's note: Not included is the important feedback between the newly developed and reusable software, and changing specifications which contribute to reuse growth and decay
Software Process Improvement (SPI)	Major feedback loops between process improvements, process maturity, personnel mix and project cycle time.
Internet RAD	Early error detection and rework. Short-term "instant" feedback on bugs from the internet site. Long-term customer feedback on desired changes and capabilities after "personalization" of the site.

References

[1] *CS599 Software Process Modeling* main web page (includes full project reports and other system dynamics links), http://sunset.usc.edu/classes/cs599_99

[2] Madachy R, *A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment*, Ph.D. Dissertation, Dept. of Industrial and Systems Engineering, University of Southern California, December 1994

[3] Madachy R, *Tutorial: Cost/Schedule/Process Modeling via System Dynamics*, Proceedings of the 14th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA, October 1999

[4] Forthcoming book: Madachy R, Boehm B, *Software Process Dynamics*, IEEE Computer Society Press, Washington, D.C., 2001, http://sunset.usc.edu/Research_Group/ray/spd

[5] Burke S, *Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization*, CMU/SEI-96-TR-024, 1997